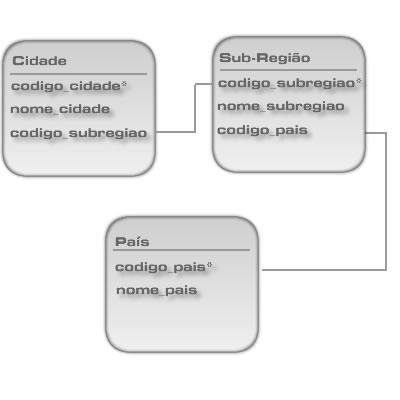
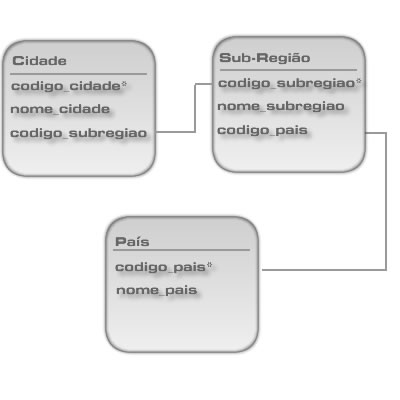
SENAC

Aula Banco de Dados I

Para os exemplos se utilizará uma estrutura de três tabelas simples com alguns dados inseridos. O diagrama abaixo representa o relacionamento entre as tabelas:





|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Tabela Cidade** | | | | | |  | |
| **codigo** | | | **cidade** | | **subregiao** |  | |
| 1 | | | Curitiba | | 1 |  | |
| 2 | | | Sao Paulo | | 2 |  | |
| 3 | | | Guarulhos | | 2 |  | |
| 4 | | | Buenos Aires | | 4 |  | |
| 5 | | | La Plata | | 4 |  | |
| 6 | | | Cordoba | | 5 |  | |
| 7 | | | Los Angeles | | 6 |  | |
| 8 | | | San Francisco | | 6 |  | |
| 9 | | | Orlando | | 7 |  | |
| 10 | | | Miami | | 7 |  | |
| 11 | | | Siena | | 8 |  | |
| 12 | | | Florenca | | 8 |  | |
| 13 | | | Milao | | 9 |  | |
| 14 | | | Yokohama | | Null |  | |
| **Tabela Subregiao** | | | | | |  | |
| **codigo** | **subregiao** | | | **pais** | |  | |
| 1 | Parana | | | 1 | |  | |
| 2 | Sao Paulo | | | 1 | |  | |
| 3 | Rio Grande do Sul | | | 1 | |  | |
| 4 | Buenos Aires | | | 2 | |  | |
| 5 | Cordoba | | | 2 | |  | |
| 6 | California | | | 3 | |  | |
| 7 | Florida | | | 3 | |  | |
| 8 | Toscana | | | 4 | |  | |
| 9 | Lombardia | | | 4 | |  | |
| 10 | Aquitania | | | 5 | |  | |
| 11 | Borgonha | | | 5 | |  | |
| 12 | Calabria | | | 5 | |  | |
| 13 | Massachussetts | | | 3 | |  | |
| 14 | Chiapas | | | Null | |  | |
| **Tabela País** | | | | | |  | |
| **codigo** | | **pais** | | | |  | |
| 1 | | Brasil | | | |  | |
| 2 | | Argentina | | | |  | |
| 3 | | Estados Unidos | | | |  | |
| 4 | | Italia | | | |  | |
| 5 | | Franca | | | |  | |
| 6 | | Noruega | | | |  | |
| **Script SQL para criação das tabelas** | | | | | | | |
| **Tabela cidade** | | | | | | | |
| CREATE TABLE "public"."cidade" ( "codigo\_cidade" SERIAL,  "nome\_cidade" VARCHAR(50),  "codigo\_subregiao" INTEGER,  CONSTRAINT "cidade\_pkey" PRIMARY KEY("codigo\_cidade") ) WITH OIDS; | | | | | | | |
| **Tabela subregiao** | | | | | | | |
| CREATE TABLE "public"."subregiao" ( "codigo\_subregiao" SERIAL,  "nome\_subregiao" VARCHAR(50),  "codigo\_pais" INTEGER,  CONSTRAINT "subregiao\_pkey" PRIMARY KEY("codigo\_subregiao") ) WITH OIDS; | | | | | | | |
| **Tabela país** | | | | | | | |
| CREATE TABLE "public"."pais" ( "codigo\_pais" SERIAL,  "nome\_pais" VARCHAR(50),  CONSTRAINT "pais\_pkey" PRIMARY KEY("codigo\_pais") ) WITH OIDS; | | | | | | | |
| **Inserção de dados** | | | | | | | |
| INSERT INTO pais (nome\_pais) VALUES ('Brasil'); INSERT INTO pais (nome\_pais) VALUES ('Argentina'); INSERT INTO pais (nome\_pais) VALUES ('Estados Unidos'); INSERT INTO pais (nome\_pais) VALUES ('Italia'); INSERT INTO pais (nome\_pais) VALUES ('Franca'); INSERT INTO pais (nome\_pais) VALUES ('Noruega');  INSERT INTO subregiao ( nome\_subregiao, codigo\_pais) VALUES ( 'Parana', 1); INSERT INTO subregiao ( nome\_subregiao, codigo\_pais) VALUES ( 'Sao Paulo', 1); INSERT INTO subregiao ( nome\_subregiao, codigo\_pais) VALUES ( 'Rio Grande do Sul', 1); INSERT INTO subregiao ( nome\_subregiao, codigo\_pais) VALUES ( 'Buenos Aires', 2); INSERT INTO subregiao ( nome\_subregiao, codigo\_pais) VALUES ( 'Cordoba', 2); INSERT INTO subregiao ( nome\_subregiao, codigo\_pais) VALUES ( 'California', 3); INSERT INTO subregiao ( nome\_subregiao, codigo\_pais) VALUES ( 'Florida', 3); INSERT INTO subregiao ( nome\_subregiao, codigo\_pais) VALUES ( 'Toscana', 4); INSERT INTO subregiao ( nome\_subregiao, codigo\_pais) VALUES ( 'Lombardia', 4); INSERT INTO subregiao ( nome\_subregiao, codigo\_pais) VALUES ( 'Aquitania', 5); INSERT INTO subregiao ( nome\_subregiao, codigo\_pais) VALUES ( 'Borgonha', 5); INSERT INTO subregiao ( nome\_subregiao, codigo\_pais) VALUES ( 'Calabria', 5); INSERT INTO subregiao ( nome\_subregiao, codigo\_pais) VALUES ( 'Massachussetts', 3); INSERT INTO subregiao ( nome\_subregiao, codigo\_pais) VALUES ( 'Chiapas', NULL);  INSERT INTO cidade (nome\_cidade, codigo\_subregiao) VALUES ('Curitiba', 1); INSERT INTO cidade (nome\_cidade, codigo\_subregiao) VALUES ('Sao Paulo', 2); INSERT INTO cidade (nome\_cidade, codigo\_subregiao) VALUES ('Guarulhos', 2); INSERT INTO cidade (nome\_cidade, codigo\_subregiao) VALUES ('Buenos Aires', 4); INSERT INTO cidade (nome\_cidade, codigo\_subregiao) VALUES ('La Plata', 4); INSERT INTO cidade (nome\_cidade, codigo\_subregiao) VALUES ('Cordoba', 5); INSERT INTO cidade (nome\_cidade, codigo\_subregiao) VALUES ('Los Angeles', 6); INSERT INTO cidade (nome\_cidade, codigo\_subregiao) VALUES ('San Francisco', 6); INSERT INTO cidade (nome\_cidade, codigo\_subregiao) VALUES ('Orlando', 7); INSERT INTO cidade (nome\_cidade, codigo\_subregiao) VALUES ('Miami', 7); INSERT INTO cidade (nome\_cidade, codigo\_subregiao) VALUES ('Siena', 8); INSERT INTO cidade (nome\_cidade, codigo\_subregiao) VALUES ('Florenca', 8); INSERT INTO cidade (nome\_cidade, codigo\_subregiao) VALUES ('Milao', 9); INSERT INTO cidade (nome\_cidade, codigo\_subregiao) VALUES ('Yokohama', NULL); | | | | | | | |

A junção de tabelas ocasiona uma tabela derivada de outras duas tabelas (reais ou derivadas), de acordo com as regras do tipo de junção. No PostgreSQL as junções são classificadas como sendo qualificadas ou cruzadas.

**Junções cruzadas**

SELECT \* FROM Tabela1 CROSS JOIN Tabela2  
  
Cada linha de Tabela1 irá combinar-se com todas as linhas de Tabelas2. Para cada combinação de linhas de Tabela1 e Tabela2, a tabela derivada conterá uma linha com todas as colunas de Tabela1 seguidas por todas as colunas de Tabela2. O número de linhas retornadas por esta consulta sempre será o número de linhas de Tabela1 multiplicado pelo número de linha de Tabela2. Por exemplo, se Tabela1 possuir 20 linhas e Tabela2 possuir 10 linhas, será retornado 200 linhas. A consulta SELECT \* FROM cidade CROSS JOIN subregiao de nosso exemplo retornará 196 linhas.

Destas 196 linhas retornadas a maioria pode ser considerada inútil, portanto, deve-se selecionar os nossos dados através de condições para nossa consulta. Essas condições são adicionadas através de cláusula WHERE.

SELECT \* FROM cidade CROSS JOIN subregiao WHERE cidade.codigo\_subregiao = subregiao.codigo\_subregiao

O uso de CROSS JOIN permite a junção de apenas duas tabelas. No entanto, nosso exemplo precisa juntar três tabelas, para isso, teremos que primeiro unir duas tabelas, para que o resultado desta junção seja utilizado com a terceira tabela.

SELECT \* FROM cidade CROSS JOIN (subregiao CROSS JOIN pais).

Utilizar SELECT \* FROM cidade CROSS JOIN subregiao equivale a utilizar SELECT \* FROM cidade, subregiao, tanto uma como outra retornará as mesmas 196 linhas e utilizar SELECT \* FROM cidade CROSS JOIN (subregiao CROSS JOIN pais) equivale a SELECT \* FROM cidade, subregiao, pais, ambas retornarão as mesmas 1176 linhas.

SELECT \* FROM cidade, subregiao where cidade.codigo\_subregiao = subregiao.codigo\_subregiao

**Junções Qualificadas**

As junções qualificadas trazem um pouquinho mais de complexidade e são divididas em junções internas e externas. Na utilização de junção qualificada, se não for especificado como junção interna ou externa, por padrão o PostgreSQL considera como sendo interna.

**Junções internas**

A utilização da cláusula INNER é o que caracteriza o comando para uma junção interna, porém, ele não é obrigatório. Pode parecer à primeira vista que as junções internas se equiparam com as junções cruzadas vistas anteriormente, até por que as duas consultas a seguir são equivalentes:

SELECT \* FROM cidade CROSS JOIN subregiao

SELECT \* FROM cidade INNER JOIN subregiao ON TRUE

Mas nas junções internas é sempre obrigatória a especificação de condição de junção, ou seja, quais linhas de uma tabela têm alguma ligação com a linha de outra tabela. Para isso podemos utilizar uma das cláusulas ON ou USING ou utilizar a palavra NATURAL no nosso comando.

A cláusula ON é o mais comumente utilizado por se assemelhar com a cláusula WHERE, ou seja, um par de linhas de Tabela1 e Tabela2 são correspondentes, se a expressão da cláusula ON produz um resultado verdade (*true*) para este par de linhas.

SELECT \* FROM cidade INNER JOIN subregiao ON  
cidade.codigo\_subregiao = subregiao.codigo\_subregiao

SELECT \* FROM cidade, subregiao where cidade.codigo\_subregiao = subregiao.codigo\_subregiao

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| codigo\_ cidade | nome\_ cidade | codigo\_ subregiao | codigo\_ subregiao\_1 | nome\_subregiao | codigo\_pais |
| 1 | Curitiba | 1 | 1 | Parana | 1 |
| 2 | Sao Paulo | 2 | 2 | Sao Paulo | 1 |
| 3 | Guarulhos | 2 | 2 | Sao Paulo | 1 |
| 4 | Buenos Aires | 4 | 4 | Buenos Aires | 2 |
| 5 | La Plata | 4 | 4 | Buenos Aires | 2 |
| 6 | Cordoba | 5 | 5 | Cordoba | 2 |
| 8 | San Francisco | 6 | 6 | California | 3 |
| 7 | Los Angeles | 6 | 6 | California | 3 |
| 9 | Orlando | 7 | 7 | Florida | 3 |
| 10 | Miami | 7 | 7 | Florida | 3 |
| 11 | Siena | 8 | 8 | Toscana | 4 |
| 12 | Florenca | 8 | 8 | Toscana | 4 |
| 13 | Milao | 9 | 9 | Lombardia | 4 |

A cláusula USING traz alguma semelhança com o ON, por também retornar um valor verdadeiro ou falso para aquele conjunto de linhas, no entanto, ele é uma forma mais rápida e abreviada de criação da consulta. Passando um nome de coluna, a execução desta consulta irá procurar nas tabelas a coluna especificada e comparar as duas. Por exemplo, t1 INNER JOIN t2 USING (a, b, c) equivale a t1 INNER JOIN t2 ON (t1.a = t2.a AND t1.b = t2.b AND t1.c = t2.c). Portanto, a consulta anterior equivale à consulta abaixo:

SELECT \* FROM subregiao INNER JOIN cidade USING (codigo\_subregiao)

Para facilitar mais, existe a utilização de NATURAL, que nada mais é abreviação de USING. Com NATURAL, a consulta encontrará todas as colunas que tem nomes iguais nas duas tabelas e fará a comparação de igualdade. O exemplo de USING acima equivale ao seguinte:

SELECT \* FROM subregiao NATURAL INNER JOIN cidade

Mas cuidado com a utilização de NATURAL, pois, ele vai comparar todas as colunas com nomes iguais, o que pode trazer resultados inesperados quando houver duas colunas com o mesmo nome e estas não tenham nenhuma relação.

Compare

SELECT \* FROM subregiao NATURAL INNER JOIN cidade

where nome\_subregiao = 'California'

**Junções Externas**

Para representar uma junção externa utiliza-se a cláusula OUTER, no entanto, ela não é obrigatória. O que caracteriza realmente as junções externas são as cláusulas LEFT, RIGHT e FULL. As cláusulas ON, USING e NATURAL valem da mesma forma nas junções internas e externas.

LEFT OUTER JOIN  
  
Primeiro, uma junção interna é realizada. Depois, para cada linha de T1 que não satisfaz a condição de junção com nenhuma linha de T2, uma linha juntada é adicionada com valores nulos nas colunas de T2. Portanto, a tabela juntada possui, incondicionalmente, no mínimo uma linha para cada linha de  
T1.

SELECT \* FROM subregiao LEFT OUTER JOIN cidade USING (codigo\_subregiao)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| codigo\_subregiao | nome\_subregiao | codigo\_pais | codigo\_cidade | nome\_cidade |
| 1 | Parana | 1 | 1 | Curitiba |
| 2 | Sao Paulo | 1 | 2 | Sao Paulo |
| 2 | Sao Paulo | 1 | 3 | Guarulhos |
| 3 | Rio G. do Sul | 1 | Null | Null |
| 4 | Buenos Aires | 2 | 4 | Buenos Aires |
| 4 | Buenos Aires | 2 | 5 | La Plata |
| 5 | Cordoba | 2 | 6 | Cordoba |
| 6 | California | 3 | 7 | Los Angeles |
| 6 | California | 3 | 8 | San Francisco |
| 7 | Florida | 3 | 9 | Orlando |
| 7 | Florida | 3 | 10 | Miami |
| 8 | Toscana | 4 | 11 | Siena |
| 8 | Toscana | 4 | 12 | Florenca |
| 9 | Lombardia | 4 | 13 | Milao |
| 10 | Aquitania | 5 | Null | Null |
| 11 | Borgonha | 5 | Null | Null |
| 12 | Calabria | 5 | Null | Null |
| 13 | Massachussetts | 3 | Null | Null |

Reparem nas linhas destacadass acima. As sub-regiões Rio Grande do Sul, Aquitania, Borgonha, Calabria e Massachussets não possuem nenhuma cidade registrada. Em uma consulta normal eles seriam ignorados. Com o uso de LEFT todos as linhas das tabelas da esquerda que não possuem correspondentes na tabela da direita são acrescidas no resultado da consulta.

RIGHT OUTER JOIN

Primeiro, uma junção interna é realizada. Depois, para cada linha de T2 que não satisfaz a condição de junção com nenhuma linha de T1, uma linha juntada é adicionada com valores nulos nas colunas de T1. É o oposto da junção esquerda: a tabela resultante possui, incondicionalmente, uma linha para cada linha de T2.

SELECT \* FROM subregiao RIGHT OUTER JOIN pais USING (codigo\_pais)

|  |  |  |  |
| --- | --- | --- | --- |
| codigo\_pais | codigo\_subregiao | nome\_subregiao | nome\_cidade |
| 1 | 2 | Sao Paulo | Brasil |
| 1 | 3 | Rio Grande do Sul | Brasil |
| 1 | 1 | Parana | Brasil |
| 2 | 4 | Buenos Aires | Argentina |
| 2 | 5 | Cordoba | Argentina |
| 3 | 13 | Massachussetts | Estados Unidos |
| 3 | 6 | California | Estados Unidos |
| 3 | 7 | Florida | Estados Unidos |
| 4 | 9 | Lombardia | Italia |
| 4 | 8 | Toscana | Italia |
| 5 | 10 | Aquitania | Franca |
| 5 | 11 | Borgonha | Franca |
| 5 | 12 | Calabria | Franca |
| 6 | Null | Null | Noruega |

Basicamente, a diferença entre RIGHT e LEFT está na escolha da tabela em que os elementos que não possuem correspondentes serão escolhidos para ser acrescidos no resultado da consulta. Neste exemplo, Noruega não tem nenhuma sub-região cadastrada, mas mesmo assim ele entra no resultado final.

FULL OUTER JOIN

Primeiro, uma junção interna é realizada. Depois, para cada linha de T1 que não satisfaz a condição de junção com nenhuma linha de T2, uma linha juntada é adicionada com valores nulos nas colunas de T2. Também, para cada linha de T2 que não satisfaz a condição de junção com nenhuma linha de T1, uma linha juntada com valores nulos nas colunas de T1 é adicionada.

SELECT \* FROM subregiao FULL OUTER JOIN cidade USING (codigo\_subregiao)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| codigo\_subregiao | nome\_subregiao | codigo\_pais | codigo\_cidade | nome\_cidade |
| 1 | Parana | 1 | 1 | Curitiba |
| 2 | Sao Paulo | 1 | 2 | Sao Paulo |
| 2 | Sao Paulo | 1 | 3 | Guarulhos |
| 3 | Rio G. do Sul | 1 | Null | Null |
| 4 | Buenos Aires | 2 | 4 | Buenos Aires |
| 4 | Buenos Aires | 2 | 5 | La Plata |
| 5 | Cordoba | 2 | 6 | Cordoba |
| 6 | California | 3 | 8 | San Francisco |
| 6 | California | 3 | 7 | Los Angeles |
| 7 | Florida | 3 | 9 | Orlando |
| 7 | Florida | 3 | 10 | Miami |
| 8 | Toscana | 4 | 11 | Siena |
| 8 | Toscana | 4 | 12 | Florenca |
| 9 | Lombardia | 4 | 13 | Milao |
| 10 | Aquitania | 5 | Null | Null |
| 11 | Borgonha | 5 | Null | Null |
| 12 | Calabria | 5 | Null | Null |
| 13 | Massachussetts | 3 | Null | Null |
| Null | Null | Null | 14 | Yokohama |

O uso de FULL não é nada mais que a utilização de RIGHT e LEFT juntos. Neste exemplo, foram acrescidos 5 sub-regiões que não posuem nenhum correspondente na tabela de cidade e nesta consulta apareceu a cidade de Yokohama que não possui uma sub-região.

Para buscarmos todos os dados de nossas tabelas utilizando JOIN podemos usar o seguinte comando:

SELECT \* FROM cidade FULL JOIN (subregiao FULL JOIN pais USING (codigo\_pais)) USING (codigo\_subregiao)

ou

SELECT \* FROM cidade FULL JOIN subregiao FULL JOIN pais USING (codigo\_pais) USING (codigo\_subregiao)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| codigo\_ subregiao | codigo\_ cidade | nome\_cidade | codigo\_ pais | nome\_subregiao | nome\_pais |
| 1 | 1 | Curitiba | 1 | Parana | Brasil |
| 2 | 2 | Sao Paulo | 1 | Sao Paulo | Brasil |
| 2 | 3 | Guarulhos | 1 | Sao Paulo | Brasil |
| 3 | Null | Null | 1 | Rio G. do Sul | Brasil |
| 4 | 4 | Buenos Aires | 2 | Buenos Aires | Argentina |
| 4 | 5 | La Plata | 2 | Buenos Aires | Argentina |
| 5 | 6 | Cordoba | 2 | Cordoba | Argentina |
| 6 | 8 | San Francisco | 3 | California | Estados Unidos |
| 6 | 7 | Los Angeles | 3 | California | Estados Unidos |
| 7 | 9 | Orlando | 3 | Florida | Estados Unidos |
| 7 | 10 | Miami | 3 | Florida | Estados Unidos |
| 8 | 11 | Siena | 4 | Toscana | Itália |
| 8 | 12 | Florenca | 4 | Toscana | Itália |
| 9 | 13 | Milao | 4 | Lombardia | Itália |
| Null | 14 | Yokohama | Null | Null | Null |
| 10 | Null | Null | 5 | Aquitania | Franca |
| 11 | Null | Null | 5 | Borgonha | Franca |
| 12 | Null | Null | 5 | Calabria | Franca |
| 13 | Null | Null | 3 | Massachussets | Estados Unidos |
| 14 | Null | Null | Null | Chiapas | Null |
| Null | Null | Null | 6 | Null | Noruega |

Compare

**SELECT \* FROM cidade FULL JOIN (subregiao FULL JOIN pais USING (codigo\_pais))**

**USING (codigo\_subregiao) where nome\_subregiao = 'Florida'**

**Atenção nas condições da consulta**

Quando se usa junção externa deve-se ter muito cuidado com as condições utilizadas na consulta, pois, lembre-se que nestas consultas mesmo que a condição não satisfaça uma linha em comparação com linhas da outra tabela, elas serão retornadas acompanhadas de valores nulos. Vejam um exemplo:

Se acaso quiser saber quais são as cidades registradas como sendo da região de Toscana, as consultas abaixo podem não ser as mais apropriadas:

SELECT cidade.descricao, subregiao.descricao FROM cidade LEFT OUTER JOIN subregiao ON cidade.codigo\_subregiao = subregiao.codigo\_subregiao AND subregiao.descricao = "Toscana"

|  |  |
| --- | --- |
| descricao | descricao\_1 |
| Curitiba | Null |
| Sao Paulo | Null |
| Guarulhos | Null |
| Buenos Aires | Null |
| La Plata | Null |
| Cordoba | Null |
| San Francisco | Null |
| Los Angeles | Null |
| Orlando | Null |
| Miami | Null |
| Siena | Toscana |
| Florenca | Toscana |
| Milao | Null |
| Yokohama | Null |

SELECT nome\_cidade, nome\_subregiao FROM cidade RIGHT OUTER JOIN subregiao

ON cidade.codigo\_subregiao = subregiao.codigo\_subregiao AND nome\_subregiao = 'Toscana'

|  |  |
| --- | --- |
| descricao | descricao\_1 |
| Null | Parana |
| Null | Sao Paulo |
| Null | Rio Grande do Sul |
| Null | Buenos Aires |
| Null | Cordoba |
| Null | California |
| Null | Florida |
| Siena | Toscana |
| Florenca | Toscana |
| Null | Lombardia |
| Null | Aquitania |
| Null | Borgonha |
| Null | Calabria |
| Null | Massachussetts |
| Null | Chiapas |

Compare

SELECT nome\_cidade, nome\_subregiao FROM cidade **LEFT** OUTER JOIN subregiao

ON cidade.codigo\_subregiao = subregiao.codigo\_subregiao AND nome\_subregiao = 'Toscana'

O mais correto para esta consulta é utilizar

SELECT nome\_cidade, nome\_subregiao FROM cidade INNER JOIN subregiao

USING (codigo\_subregiao) WHERE nome\_subregiao = 'Toscana'

ou

SELECT nome\_cidade, nome\_subregiao FROM cidade INNER JOIN subregiao

ON cidade.codigo\_subregiao = subregiao.codigo\_subregiao AND nome\_subregiao = 'Toscana'

que retornarão o mesmo resultado:

|  |  |
| --- | --- |
| nome\_cidade | nome\_subregiao |
| Siena | Toscana |
| Florenca | Toscana |